AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions of claims in the application:

Listing of Claims:

1. (Currently Amended) A system embodied on a computer-readable storage medium that facilitates extending the functionality of an application, comprising:

a schema component that includes a schema element representative of domain terminology of a problem to be solved in a host application, the domain terminology is not native to a general application programming interface (API) of the host application; and

a mapping component that maps the schema element to a construct of ~~a host~~ an API ~~application programmable interface (API)~~ of the host application and enables the host application to operate on the domain terminology.

2. (Currently Amended) The system of claim 1, the ~~API~~ host application is ~~associated with~~ at least one of a word processing application, spreadsheet application, drawing application, presentation graphics application, website design and development application, database application, project management application, publication application, note management application ~~and,~~ or browser and communication application.

3. (Original) The system of claim 1, the schema component facilitates generation of at least one of a data programming model and a view programming model.

4. (Original) The system of claim 1, further comprising a generation component that generates at least one of a data programming model and a view programming model.

5. (Currently Amended) The system of claim 1, further comprising a generation component that generates a data programming model and a view programming model that are automatically connected to each other *via* data binding, the view programming model provides an interface by which the host application operates on the domain terminology.

6. (Original) The system of claim 1, further comprising a generation component that generates at least one of a data programming model and a view programming model, wherein the data programming model interfaces to the host API *via* the view programming model.

7. (Original) The system of claim 1, further comprising a separation component that separates data from document content.

8. (Original) The system of claim 1, further comprising a separation component that generates a data island in a document of a host application.

9. (Original) The system of claim 8, the data island is editable.

10. (Currently Amended) The system of claim 8, the data island can be at least one of accessed ~~and~~ or modified without launching the host application.

11. (Original) The system of claim 8, contents of the data island and contents of the document are synchronized when the document is run inside the host application *via* data binding.

12. (Currently Amended) The system of claim 1, the schema component and the mapping component facilitate generation of a new API that interfaces to the host API and enables the host application to operate on the domain terminology.

13. (Cancelled).

14. (Currently Amended) A computer, comprising: ~~that employs the system of claim 1~~

a schema component that includes a schema element representative of domain terminology of a problem to be solved in a host application stored at the storage media, the domain terminology is not part of a native API of the host application; and

a mapping component that maps the schema element to a construct of an API of the host application and enables the host application to operate on the domain terminology;

processing hardware to execute instructions associated with the schema component and mapping component.

15. (Currently Amended) The system of claim 1, the schema component includes a view schema that represents only data of interest of [[a]] the host application.

16. (Currently Amended) The system of claim 1, the schema component includes a view schema that facilitates pulling a plurality of objects of interest from a plurality of ~~the host~~ APIs of the host application.

17. (Currently Amended) The system of claim 1, at least one of the schema component and the mapping component facilitate generation of a view API that is a hybrid of view schema and the ~~host~~ API of the host application.

18. (Currently Amended) A system embodied on computer-readable storage media that facilitates extending the functionality of an application, comprising:

a schema component that includes a schema in terms of a problem to be solved in a host application that are not native to the host application and a mapping of the terms to generic objects of an API of [[a]] the host application; and

a generation component that generates at least one programming model based on the schema that interfaces to the API.

19. (Original) The system of claim 18, further comprising a separation component that generates an editable data island in a document of the host application.

20. (Original) The system of claim 19, contents of the data island and contents of the document are synchronized when the document is run inside the host application.

21. (Original) The system of claim 18, the schema component facilitates generation of a new API that interfaces to the API.

22. (Original) The system of claim 18, the schema component facilitates manipulation of a variable without reference to underlying register and stack allocations.

23. (Currently Amended) The system of claim 18, further comprising a separation component that generates an editable data island in a document of [[a]] the host application, which data island is accessible and modifiable without running the host application.

24. (Original) A method of extending the functionality of an application, comprising:

creating a schema of problem domain elements of a problem to be solved, the domain elements are not native to an API of the application;

mapping the problem domain elements to constructs interpretable by one or more generic application interfaces of the application; and

generating a program model based on the mapping of the problem domain elements such that the one or more generic application interfaces can be accessed *via* the program model using the problem domain elements.

25. (Original) The method of claim 24, further comprising automatically separating the program model into a data model and a view model.

26. (Original) The method of claim 24, further comprising exposing data of the problem domain elements as named objects in a view model.

27. (Original) The method of claim 24, further comprising exposing data of the problem domain elements as declarations in a data model.

28. (Original) The method of claim 24, the program model is a schema-based, machine generated model.

29. (Original) The method of claim 24, further comprising exposing data of the problem domain elements as first class named objects.

30. (Original) The method of claim 24, further comprising separating data from a view model of the program model by,

generating data that conforms to the schema; and

saving the data as a data island in a document of the application.

31. (Currently Amended) A computer-readable storage medium having stored thereon computer-executable instructions for performing a method for extending the functionality of an application, ~~the method~~ comprising:

creating a mapping of a schema to one or more generic application interfaces of the application;

generating a view model from the mapped schema, the view model includes view data that is mapped to objects of the application, the view data is not native to the generic application interfaces of the application; and

generating a data model from the mapped schema, the data model including data that is mapped to objects of the application.

32. (Currently Amended) The ~~method~~ computer-readable storage medium of claim 31, further comprising data binding the view model to the data model.

33. (Currently Amended) The ~~method~~ computer-readable storage medium of claim 31, the view model uses portions of the schema that are mapped to problem domain terms related to a problem to be solved.

34. (Currently Amended) The ~~method~~ computer-readable storage medium of claim 31, further comprising extracting the view data that is mapped to objects of the applications and exposing the view data as view controls.

35. (Currently Amended) The ~~method~~ computer-readable storage medium of claim 31, the data model is generated by,

conforming the data to the schema; and

saving the data as a data island in a document of the application.

36. (Currently Amended) The ~~method~~ computer-readable storage medium of claim 35, further comprising:

connecting the data model to the data island; and

synchronizing contents of the data island with contents of the document when the application processes the document.

37. (Currently Amended) A system embodied on computer-readable storage media that facilitates extending the functionality of an application comprising:

means for creating a mapping of a schema to one or more generic application interfaces of the application;

means for generating a view model from the mapped schema, the view model includes view data that is mapped to objects of the application, the view data is not native to the generic application interfaces of the application;

means for generating a data model from the mapped schema, the data model including data that is mapped to objects of the application; and

means for propagating changes to the data model to contents of a document *via* a data binding mechanism to view controls.